Security Properties of Self-similar Uniformly Parameterised Systems of Cooperations

Peter Ochsenschläger and Roland Rieke
Fraunhofer Institute for Secure Information Technology, SIT
Darmstadt, Germany
Email: peter-ochsenschlaeger@t-online.de, roland.rieke@sit.fraunhofer.de

***Abstract*—Uniform parameterisations of cooperations are defined in terms of formal language theory, such that each pair of partners cooperates in the same manner, and that the mechanism (schedule) to determine how one partner may be involved in several cooperations, is the same for each partner. Generalising each pair of partners cooperating in the same manner, for such systems of cooperations a kind of self-similarity is formalised. From an abstracting point of view, where only actions of some selected partners are considered, the complex system of all partners behaves like the smaller subsystem of the selected partners. For verification purposes, so called uniformly parameterised safety properties are defined. Such properties can be used to express privacy policies as well as security and dependability requirements. It is shown, how the parameterised problem of verifying such a property is reduced by self-similarity to a finite state problem.**

***Keywords*-cooperations as prefix closed languages; abstractions of system behaviour; self-similarity in systems of cooperations; privacy policies; uniformly parameterised safety properties;**

I. Introduction

As an example for cooperations let us consider an e-commerce protocol, that determines how two cooperation partners have to perform a certain kind of financial transactions. As such a protocol should work for several partners in the same manner, it is parameterised by the partners and the parameterisation should be uniform w.r.t. the partners. It is quite evident that similar requirements have to be fulfilled in any highly scalable system or system of systems such as cloud computing platforms or vehicular communication systems in which vehicles and roadside units communicate in ad hoc manner to exchange traffic information [1].

In this paper (Sect. III) we formalise uniform parameterisations of two-sided cooperations in terms of formal language theory, such that each pair of partners cooperates in the same manner, and that the mechanism (schedule) to determine how one partner may be involved in several cooperations, is the same for each partner. Generalising each pair of partners cooperating in the same manner, the following kind of self-similarity is desirable for such systems of cooperations: From an abstracting point of view, where only actions of some selected partners are considered, the complex system of all partners behaves like the smaller subsystem of the selected partners (Sect. IV).

For verification purposes it is of interest to know, which kind of dynamic system properties are "compatible" with self-similarity. Therefore in Sect. V so called uniformly parameterised safety properties are defined. An example shows how such properties can be used to express privacy policies. Subsequently, it is shown how the parameterised problem of verifying such a property is reduced by self-similarity to a finite state problem under certain regularity restrictions. Liveness aspects of self-similar systems will be subject of a forthcoming paper (see Sect. VI).

II. Related Work

Verification approaches for parameterised systems.: An extension to the *Mur*φ verifier to verify systems with replicated identical components through a new data type called RepetitiveID is presented in [2]. During the verification Murφ checks if the result can be generalised for a family of systems. The soundness of the abstraction algorithm is guaranteed by the restrictions on the use of repetitiveIDs. A typical application area of this tool are cache coherence protocols. The aim of [3] is an abstraction method through symmetry which works also when using variables holding references to other processes which is not possible in Murφ. An implementation of this approach for the *SPIN* model-checker (http://spinroot.com/) is described. In [4] a methodology for constructing abstractions and refining them by analysing counter-examples is presented. The method combines abstraction, model-checking and deductive verification and in particular, allows to use the set of reachable states of the abstract system in a deductive proof even when the abstract model does not satisfy the specification and when it simulates the concrete system with respect to a weaker simulation notion than Milner's. The tool *InVeSt* supports this approach and makes use of *PVS* (http://pvs.csl.sri.com/) and *SMV* (http://www.cs.cmu.edu/~modelcheck/smv.html). This approach however does not consider liveness properties. In [5] a technique for automatic verification of parameterised systems based on process algebra *CCS* [6] and the logic modal *mu-calculus* [7] is presented. This technique views processes as property transformers and is based on computing the limit of a sequence of mu-calculus formula generated

by these transformers. In [8] we developed an *abstraction based approach* to extend our tool supported verification techniques to be able to verify families of parameterised systems, independent of the exact number of replicated components. The above-mentioned approaches demonstrate, that finite state methods combined with deductive methods can be applied to analyse parameterised systems. The approaches differ in varying amounts of user intervention and their range of application. A survey of approaches to combine model checking and theorem proving methods is given in [9].

Iterated shuffle products.: In [10] it is shown that our definition of uniformly parameterised cooperations is strongly related to iterated shuffle products [11], if the cooperations are "structured into phases". For such systems of cooperations a sufficient condition for the kind of self-similarity which we use here is given. Under certain regularity restrictions this condition can be verified by a semi-algorithm. The main concept for such a condition are shuffle automata [12] (multicounter automata [13]) whose computations, if they are deterministic, unambiguously describe how a cooperation partner is involved in several phases.

The main contribution of this paper is to show how the parameterised problem of verifying a uniformly parameterised safety property can be solved by means of the self-similarity results of [10] and finite state methods.

III. Parameterised Cooperations

The behaviour L of a discrete system can be formally described by the set of its possible sequences of actions. Therefore $L \subset \Sigma^*$ holds where Σ is the set of all actions of the system, and Σ^* (free monoid over Σ) is the set of all finite sequences of elements of Σ (words), including the empty sequence denoted by ε. Subsets of Σ^* are called formal languages. Words can be composed: if u and v are words, then uv is also a word. This operation is called the *concatenation*; especially $\varepsilon u = u\varepsilon = u$. A word u is called a *prefix* of a word v if there is a word x such that $v = ux$. The set of all prefixes of a word u is denoted by $\mathrm{pre}(u)$; $\varepsilon \in \mathrm{pre}(u)$ holds for every word u.

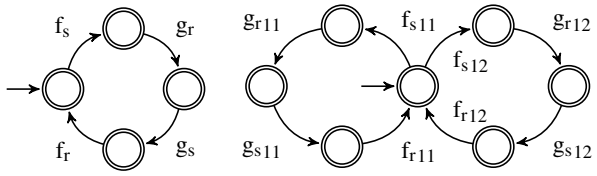
Formal languages which describe system behaviour have the characteristic that $\mathrm{pre}(u) \subset L$ holds for every word $u \in L$. Such languages are called *prefix closed*. System behaviour is thus described by prefix closed formal languages.

Different formal models of the same system are partially ordered with respect to different levels of abstraction. Formally, abstractions are described by so called alphabetic language homomorphisms. These are mappings $h^* : \Sigma^* \longrightarrow \Sigma'^*$ with $h^*(xy) = h^*(x)h^*(y)$, $h^*(\varepsilon) = \varepsilon$ and $h^*(\Sigma) \subset \Sigma' \cup \{\varepsilon\}$. So they are uniquely defined by corresponding mappings $h : \Sigma \longrightarrow \Sigma' \cup \{\varepsilon\}$. In the following we denote both the mapping h and the homomorphism h^* by h. Inverse homomorphism are denoted by h^{-1}. Let L be a language over the alphabet Σ'. Then $h^{-1}(L)$ is the set of words $w \in \Sigma^*$ such that $h(w) \in L$. In this paper we consider a lot of alphabetic language homomorphisms. So for simplicity we tacitly assume that a mapping between free monoids is an alphabetic language homomorphism if nothing contrary is stated.

To describe a two-sided cooperation, let $\Sigma = \Phi \cup \Gamma$ where Φ is the set of actions of cooperation partner F and Γ is the set of actions of cooperation partner G. Now a prefix closed language $L \subset (\Phi \cup \Gamma)^*$ formally defines a two-sided cooperation.

Example 1. *Let* $\Phi = \{\mathrm{f_s}, \mathrm{f_r}\}$ *and* $\Gamma = \{\mathrm{g_s}, \mathrm{g_r}\}$ *and hence* $\Sigma = \{\mathrm{f_s}, \mathrm{f_r}, \mathrm{g_s}, \mathrm{g_r}\}$. *An example for a cooperation* $L \subset \Sigma^*$ *is now given by the automaton in Fig. 1(a). It describes a simple handshake between F and G.*

Please note that in the following we will denote initial states by a short incoming arrow and final states by double circles. In this automaton all states are final states, since L is prefix closed.



(a) 1-1-cooperation L (b) 1-2-cooperation $\mathscr{L}_{\{1\}\{1,2\}}$

Figure 1. Automata for a simple parameterised cooperation

For parameter sets I, K and $(i,k) \in I \times K$ let Σ_{ik} denote pairwise disjoint copies of Σ. The elements of Σ_{ik} are denoted by a_{ik} and $\Sigma_{IK} := \dot{\bigcup}_{(i,k)\in I\times K} \Sigma_{ik}$. The index ik describes the bijection $a \leftrightarrow a_{ik}$ for $a \in \Sigma$ and $a_{ik} \in \Sigma_{ik}$. Now $\mathscr{L}_{IK} \subset \Sigma_{IK}^*$ (prefix-closed) describes a *parameterised system*. To avoid pathological cases we generally assume parameter and index sets to be non empty.

For a cooperation between one partner of type F with two partners of type G in Example 1 let $\Phi_{\{1\}\{1,2\}} = \{\mathrm{f_{s11}}, \mathrm{f_{r11}}, \mathrm{f_{s12}}, \mathrm{f_{r12}}\}$, $\Gamma_{\{1\}\{1,2\}} = \{\mathrm{g_{s11}}, \mathrm{g_{r11}}, \mathrm{g_{s12}}, \mathrm{g_{r12}}\}$ and $\Sigma_{\{1\}\{1,2\}} = \Phi_{\{1\}\{1,2\}} \dot\cup \Gamma_{\{1\}\{1,2\}}$. A 1-2-cooperation, where each pair of partners cooperates restricted by L and each partner has to finish the handshake it just is involved in before entering a new one, is now given (by reachability analysis) by the automaton in Fig. 1(b) for $\mathscr{L}_{\{1\}\{1,2\}}$. Fig. 2 in contrast depicts an automaton for a 2-1-cooperation $\mathscr{L}_{\{1,2\}\{1\}}$ with the same overall number of partners involved but two of type F and one partner of type G. A 3-3-cooperation with the same simple behaviour of partners already requires an automaton with 916 states and 3168 state transitions.

For $(i,k) \in I \times K$, let $\pi_{ik}^{IK} : \Sigma_{IK}^* \to \Sigma^*$ with

$$\pi_{ik}^{IK}(a_{rs}) = \begin{cases} a \mid & a_{rs} \in \Sigma_{ik} \\ \varepsilon \mid & a_{rs} \in \Sigma_{IK} \setminus \Sigma_{ik} \end{cases} .$$

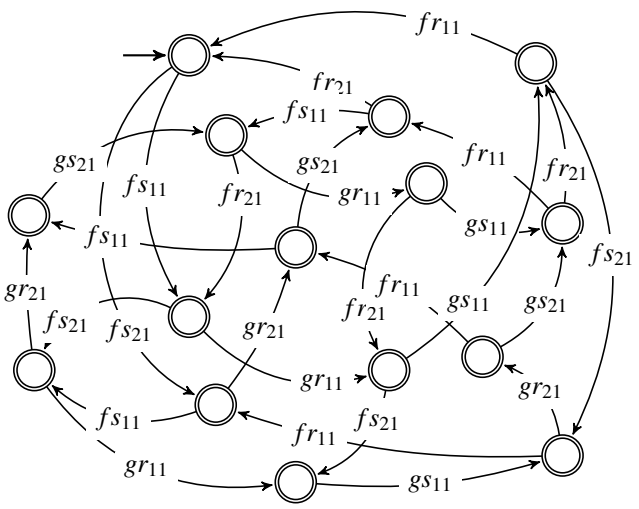


Figure 2. Automaton for the 2-1-cooperation $\mathscr{L}_{\{1,2\}\{1\}}$

For *uniformly parameterised systems* $\mathscr{L}_{IK}$ we generally want to have

$$\mathscr{L}_{IK} \subset \bigcap_{(i,k)\in I\times K} ((\pi_{ik}^{IK})^{-1}(L))$$

because from an abstracting point of view, where only the actions of a specific Σ_{ik} are considered, the complex system $\mathscr{L}_{IK}$ is restricted by L.

In addition to this inclusion $\mathscr{L}_{IK}$ is defined by *local schedules* that determine how each "version of a partner" can participate in "different cooperations". More precisely, let $SF \subset \Phi^*$, $SG \subset \Gamma^*$ be prefix closed. For $(i,k) \in I \times K$, let $\varphi_i^{IK} : \Sigma_{IK}^* \to \Phi^*$ and $\gamma_k^{IK} : \Sigma_{IK}^* \to \Gamma^*$ with

$$\varphi_i^{IK}(a_{rs}) = \left\{ \begin{array}{l|l} a & a_{rs} \in \Phi_{\{i\}K} \\ \varepsilon & a_{rs} \in \Sigma_{IK} \setminus \Phi_{\{i\}K} \end{array} \right. \text{ and}$$

$$\gamma_k^{IK}(a_{rs}) = \left\{ \begin{array}{l|l} a & a_{rs} \in \Gamma_{I\{k\}} \\ \varepsilon & a_{rs} \in \Sigma_{IK} \setminus \Gamma_{I\{k\}} \end{array} \right. ,$$

where Φ_{IK} and Γ_{IK} are defined correspondingly to Σ_{IK}.

Definition 1 (Uniformly parameterised cooperation $\mathscr{L}_{IK}$)**.** *Let I, K be finite parameter sets, then*

$$\mathscr{L}_{IK} := \bigcap_{(i,k)\in I\times K} (\pi_{ik}^{IK})^{-1}(L) \cap \\ \cap \bigcap_{i\in I} (\varphi_i^{IK})^{-1}(SF) \cap \bigcap_{k\in K} (\gamma_k^{IK})^{-1}(SG)$$

By this definition

$$\mathscr{L}_{\{1\}\{1\}} = (\pi_{11}^{\{1\}\{1\}})^{-1}(L) \cap \\ \cap\ (\varphi_1^{\{1\}\{1\}})^{-1}(SF) \cap (\gamma_1^{\{1\}\{1\}})^{-1}(SG).$$

As we want $\mathscr{L}_{\{1\}\{1\}}$ being isomorphic to L by the isomorphism $\pi_{11}^{\{1\}\{1\}} : \Sigma_{\{1\}\{1\}}^* \to \Sigma^*$ we additionally need

$$(\pi_{11}^{\{1\}\{1\}})^{-1}(L) \subset (\varphi_1^{\{1\}\{1\}})^{-1}(SF) \text{ and}$$

$$(\pi_{11}^{\{1\}\{1\}})^{-1}(L) \subset (\gamma_1^{\{1\}\{1\}})^{-1}(SG).$$

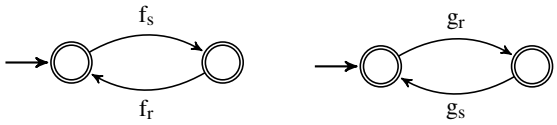
This is equivalent to $\pi_\Phi(L) \subset SF$ and $\pi_\Gamma(L) \subset SG$, where $\pi_\Phi : \Sigma^* \to \Phi^*$ and $\pi_\Gamma : \Sigma^* \to \Gamma^*$ are defined by

$$\pi_\Phi(a) = \left\{ \begin{array}{l|l} a & a \in \Phi \\ \varepsilon & a \in \Gamma \end{array} \right. \text{ and } \pi_\Gamma(a) = \left\{ \begin{array}{l|l} a & a \in \Gamma \\ \varepsilon & a \in \Phi \end{array} \right. .$$

So we complete Def. 1 by the additional conditions

$$\pi_\Phi(L) \subset SF \text{ and } \pi_\Gamma(L) \subset SG.$$

Schedules SF and SG that fit to the cooperations given in Example 1 are depicted in Figs. 3(a) and 3(b). Here we have $\pi_\Phi(L) = SF$ and $\pi_\Gamma(L) = SG$.



(a) Schedule SF (b) Schedule SG

Figure 3. Automata $\mathbb{SF}$ and $\mathbb{SG}$ for the schedules SF and SG

The system $\mathscr{L}_{IK}$ of cooperations is a typical example of a *complex system*. It consists of several identical components (copies of the two-sided cooperation L), which "interact" in a uniform manner (described by the schedules SF and SG and by the homomorphisms φ_i^{IK} and γ_k^{IK}).

Remark 1. *It is easy to see that $\mathscr{L}_{IK}$ is isomorphic to $\mathscr{L}_{I'K'}$ if I is isomorphic to I' and K is isomorphic to K'. More precisely, let $\iota_{I'}^{I} : I \to I'$ and $\iota_{K'}^{K} : K \to K'$ be bijections and let $\iota_{I'K'}^{IK} : \Sigma_{IK}^* \to \Sigma_{I'K'}^*$ be defined by*

$$\iota_{I'K'}^{IK}(a_{ik}) := a_{\iota_{I'}^{I}(i)\iota_{K'}^{K}(k)} \text{ for } a_{ik} \in \Sigma_{IK}.$$

Then $\iota_{I'K'}^{IK}$ is a isomorphism and $\iota_{I'K'}^{IK}(\mathscr{L}_{IK}) = \mathscr{L}_{I'K'}$. The set of all these isomorphisms $\iota_{I'K'}^{IK}$ defined by corresponding bijections $\iota_{I'}^{I}$ and $\iota_{K'}^{K}$ is denoted by $\mathscr{I}_{I'K'}^{IK}$.

IV. Self-similarity

By *self-similary* we want to formalise that for $I' \subset I$ and $K' \subset K$ from an abstracting point of view, where only the actions of $\Sigma_{I'K'}$ are considered, the complex system $\mathscr{L}_{IK}$ behaves like the smaller subsystem $\mathscr{L}_{I'K'}$. Therefore we now consider special abstractions on $\mathscr{L}_{IK}$.

Definition 2 (Projection abstraction)**.** *For $I' \subset I$ and $K' \subset K$ let $\Pi_{I'K'}^{IK} : \Sigma_{IK}^* \to \Sigma_{I'K'}^*$ with*

$$\Pi_{I'K'}^{IK}(a_{rs}) = \left\{ \begin{array}{l|l} a_{rs} & a_{rs} \in \Sigma_{I'K'} \\ \varepsilon & a_{rs} \in \Sigma_{IK} \setminus \Sigma_{I'K'}. \end{array} \right.$$

It is easy to see [10]:

Theorem 1. *$\mathscr{L}_{IK} \supset \mathscr{L}_{I'K'}$ for $I' \times K' \subset I \times K$, and therefore*

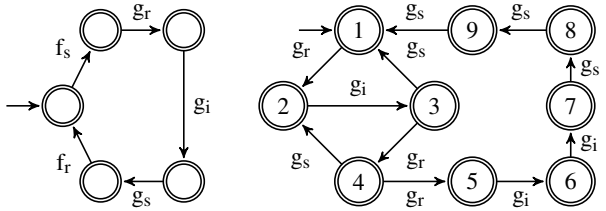
$$\Pi_{I'K'}^{IK}(\mathscr{L}_{IK}) \supset \Pi_{I'K'}^{IK}(\mathscr{L}_{I'K'}) = \mathscr{L}_{I'K'}.$$

The reverse inclusions

$$\Pi^{IK}_{I'K'}(\mathscr{L}_{IK}) \subset \mathscr{L}_{I'K'} \text{ for all } I' \times K' \subset I \times K \qquad (1)$$

do not hold in general, which is shown by the following example.

Example 2. *For a counterexample let us examine the 1-1-cooperation given by the automaton in Fig. 4(a). Let the schedule SF again be given by the automaton $\mathbb{SF}$ in Fig. 3(a) and the schedule SG be given by the automaton $\mathbb{SG}$ in Fig. 4(b).*



(a) 1-1-cooperation L (b) Schedule SG

Figure 4. Automata for the counterexample

In the automaton $\mathbb{SG}$ immediately after entering a second handshake (state 4) G may enter a third handshake but immediately after entering the first handshake (state 2) G may not enter a second handshake. We now get for example

$$f_{s11}f_{s21}f_{s31}g_{r11}g_{i11}g_{r21}g_{r31} \in \mathscr{L}_{\{1,2,3\}\{1\}}.$$

Hence

$$f_{s21}f_{s31}g_{r21}g_{r31} \in \Pi^{\{1,2,3\}\{1\}}_{\{2,3\}\{1\}}(\mathscr{L}_{\{1,2,3\}\{1\}}), \text{ but}$$

$$f_{s21}f_{s31}g_{r21}g_{r31} \notin \mathscr{L}_{\{2,3\}\{1\}}.$$

In the general case we do not know the decidability status of (1), but for many parameterised systems (1) holds, and therefore

$$\Pi^{IK}_{I'K'}(\mathscr{L}_{IK}) = \mathscr{L}_{I'K'},$$

which is a generalisation of $\pi^{IK}_{ik}(\mathscr{L}_{IK}) = L$.

Definition 3 (Self-similarity)**.**
A uniformly parameterised cooperation $\mathscr{L}_{IK}$ is called self-similar *iff*

$$\Pi^{IK}_{I'K'}(\mathscr{L}_{IK}) = \mathscr{L}_{I'K'} \text{ for each } I' \times K' \subset I \times K.$$

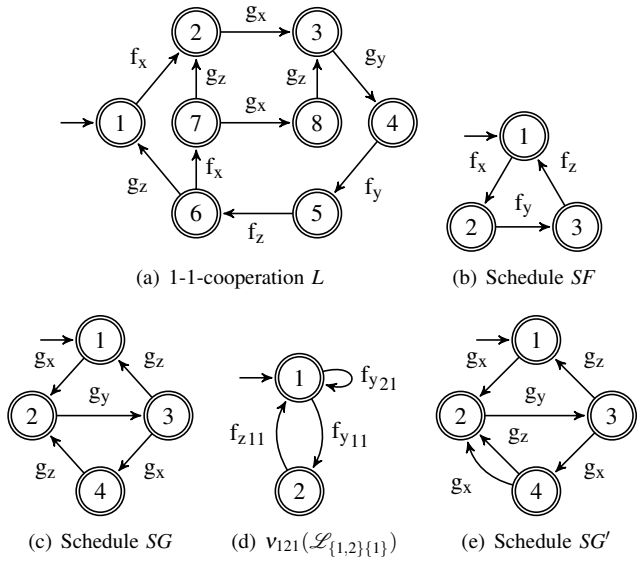
So we are looking for conditions, which imply (1). Fig. 4(b) is typical in the sense that it may serve as an idea to get a sufficient condition for self-similarity. It requires (a) two separate conditions, one for each schedule, (b) structuring schedules into phases, which may be shuffled in a restricted manner, (c) formalising "*how a cooperation partner is involved in several phases*", (d) the more phases a cooperation partner is involved in, the less possibilities of acting in each phase he has. In [10] a sufficient condition for self-similarity is given, which is based on deterministic computations in shuffle automata. Under certain regularity restrictions this condition can be verified by a semi-algorithm.

V. Uniformly Parameterised Security Properties

We will now give an example that demonstrates the significance of self-similarity for verification purposes and then present a generic verification scheme for uniformly parameterised security properties.

Example 3. *We consider a system of servers, each of them managing a resource, and clients, which want to use these resources. We assume that as a means to enforce a given* privacy policy *a server has to manage its resource in such a way that no client may access this resource during it is in use by another client (*privacy requirement*). This may be required to ensure* anonymity *in such a way that clients and their actions on a resource cannot be linked by an observer.*

We formalise this system at an abstract level, where a client may perform the actions f_x (send a request), f_y (receive a permission) and f_z (send a free-message), and a server may perform the corresponding actions g_x (receive a request), g_y (send a permission) and g_z (receive a free-message). The possible sequences of actions of a client resp. of a server are given by the automaton $\mathbb{SF}$ resp. $\mathbb{SG}$. The automaton $\mathbb{L}$ describes the 1-1-cooperation of one client and one server (see Fig. 5). These automata define the client-server system $\mathscr{L}_{IK}$.



(a) 1-1-cooperation L (b) Schedule SF

(c) Schedule SG (d) $\nu_{121}(\mathscr{L}_{\{1,2\}\{1\}})$ (e) Schedule SG'

Figure 5. Automata $\mathbb{L}$, $\mathbb{SF}$, $\mathbb{SG}$, $\nu_{121}(\mathscr{L}_{\{1,2\}\{1\}})$ and $\mathbb{SG}'$ for Example 3

Considering f_y as the begin-action and f_z as the end-action w.r.t. accessing a resource, the privacy requirement can be formalised by (2).

Let $i,i' \in I$, $i \neq i'$, $k \in K$ and
$\mu^{IK}_{ii'k} : \Sigma^*_{IK} \to \{\mathrm{f}_{y_{ik}}, \mathrm{f}_{zik}, \mathrm{f}_{y_{i'k}}\}^*$ with

$$\mu^{IK}_{ii'k}(a_{rs}) := \begin{cases} a_{rs} \mid & a_{rs} \in \{\mathrm{f}_{y_{ik}}, \mathrm{f}_{zik}, \mathrm{f}_{y_{i'k}}\} \\ \varepsilon \mid & a_{rs} \in \Sigma_{IK} \setminus \{\mathrm{f}_{y_{ik}}, \mathrm{f}_{zik}, \mathrm{f}_{y_{i'k}}\}. \end{cases}$$

For each $i,i' \in I$, $i \neq i'$ and $k \in K$

$$\mu^{IK}_{ii'k}(\mathscr{L}_{IK}) \cap \Sigma^*_{\{i,i'\}\{k\}} \mathrm{f}_{y_{ik}} \mathrm{f}_{y_{i'k}} = \emptyset. \tag{2}$$

For $i,i' \in I$, $i \neq i'$, $k \in K$ let
$\nu_{ii'k} : \Sigma^*_{\{i,i'\}\{k\}} \to \{\mathrm{f}_{y_{ik}}, \mathrm{f}_{zik}, \mathrm{f}_{y_{i'k}}\}^*$ be defined by

$$\nu_{ii'k}(a_{rs}) := \begin{cases} a_{rs} \mid & a_{rs} \in \{\mathrm{f}_{y_{ik}}, \mathrm{f}_{zik}, \mathrm{f}_{y_{i'k}}\} \\ \varepsilon \mid & a_{rs} \in \Sigma_{\{i,i'\}\{k\}} \setminus \{\mathrm{f}_{y_{ik}}, \mathrm{f}_{zik}, \mathrm{f}_{y_{i'k}}\}, \end{cases}$$

then

$$\mu^{IK}_{ii'k} = \nu_{ii'k} \circ \Pi^{IK}_{\{i,i'\}\{k\}}.$$

Hence,

$$\mu^{IK}_{ii'k}(\mathscr{L}_{IK}) = \nu_{ii'k}(\mathscr{L}_{\{i,i'\}\{k\}}) \text{ if } \mathscr{L}_{IK} \text{ is self-similar.}$$

Let $\iota_{ii'k} : \Sigma^*_{\{i,i'\}\{k\}} \to \Sigma^*_{\{1,2\}\{1\}}$ be the isomorphism defined by

$$\iota_{ii'k}(a_{ik}) := \begin{cases} a_{11} \mid & a_{ik} \in \Sigma_{ik} \\ a_{21} \mid & a_{i'k} \in \Sigma_{i'k}, \end{cases}$$

then by Remark 1

$$\iota_{ii'k}(\mathscr{L}_{\{i,i'\}\{k\}}) = \mathscr{L}_{\{1,2\}\{1\}},$$

since $\nu_{ii'k} = \iota^{-1}_{ii'k} \circ \nu_{121} \circ \iota_{ii'k}$, $\mathscr{L}_{IK}$ fulfills the privacy requirement (2) for each index set I and K iff

$$\nu_{121}(\mathscr{L}_{\{1,2\}\{1\}}) \cap \Sigma^*_{\{1,2\}\{1\}} \mathrm{f}_{y_{11}} \mathrm{f}_{y_{21}} = \emptyset. \tag{3}$$

This can be verified by checking the finite automaton of $\mathscr{L}_{\{1,2\}\{1\}}$. The automaton of $\mathscr{L}_{\{1,2\}\{1\}}$ consists of 28 states. The minimal automaton of $\nu_{121}(\mathscr{L}_{\{1,2\}\{1\}})$ is shown in Fig. 5(d) which implies (3). Self-similarity of $\mathscr{L}_{IK}$ can be shown using the methods given in [10]. So $\mathscr{L}_{IK}$ fulfills the privacy requirement.

On the contrary, $\mathscr{L}'_{IK}$ defined by SF, SG' and L of Fig. 5 is not self-similar because of

$$\mathrm{f}_{x11}\mathrm{f}_{x21}\mathrm{f}_{x31}\mathrm{g}_{x11}\mathrm{g}_{y11}\mathrm{g}_{x21}\mathrm{g}_{x31}\mathrm{g}_{y21}\mathrm{f}_{y11}\mathrm{f}_{y21} \in \mathscr{L}'_{\{1,2,3\}\{1\}},$$

$$\mathrm{f}_{x11}\mathrm{f}_{x21}\mathrm{g}_{x11}\mathrm{g}_{y11}\mathrm{g}_{x21}\mathrm{g}_{y21}\mathrm{f}_{y11}\mathrm{f}_{y21} \in \Pi^{\{1,2,3\}\{1\}}_{\{1,2\}\{1\}}(\mathscr{L}'_{\{1,2,3\}\{1\}})$$

$$\text{but } \mathrm{f}_{x11}\mathrm{f}_{x21}\mathrm{g}_{x11}\mathrm{g}_{y11}\mathrm{g}_{x21}\mathrm{g}_{y21}\mathrm{f}_{y11}\mathrm{f}_{y21} \notin \mathscr{L}'_{\{1,2\}\{1\}}.$$

The same action sequence shows that $\mathscr{L}'_{IK}$ does not fulfill the privacy requirement.

The privacy requirement of the example is a typical safety property [14]. These properties describe that "nothing forbidden happens". They can be formalised by a set $\mathscr{F}$ of forbidden action sequences. So a system $\mathscr{L}_{IK} \subset \Sigma^*_{IK}$ satisfies a safety property $\mathscr{F}_{IK} \subset \Sigma^*_{IK}$ iff $\mathscr{L}_{IK} \cap \mathscr{F}_{IK} = \emptyset$. More precisely, these are safety properties which can be expressed without "dummy actions" to descibe deadlocks.

In our example the privacy requirement is formalised by

$$\mathscr{F}_{IK} = \bigcup_{i,i' \in I, i \neq i', k \in K} (\mu^{IK}_{ii'k})^{-1}(\Sigma^*_{\{i,i'\}\{k\}} \mathrm{f}_{y_{ik}} \mathrm{f}_{y_{i'k}}) =$$

$$= \bigcup_{i,i' \in I, i \neq i', k \in K} (\Pi^{IK}_{\{i,i'\}\{k\}})^{-1}(\iota^{-1}_{ii'k}[\nu^{-1}_{121}(\Sigma^*_{\{1,2\}\{1\}} \mathrm{f}_{y_{11}} \mathrm{f}_{y_{21}})])$$

because of

$$\mu^{IK}_{ii'k} = \iota^{-1}_{ii'k} \circ \nu_{121} \circ \iota_{ii'k} \circ \Pi^{IK}_{\{i,i'\}\{k\}} \text{ and}$$

$$\iota_{ii'k}(\Sigma^*_{\{i,i'\}\{k\}} \mathrm{f}_{y_{ik}} \mathrm{f}_{y_{i'k}}) = \Sigma^*_{\{1,2\}\{1\}} \mathrm{f}_{y_{11}} \mathrm{f}_{y_{21}}.$$

As

$$\nu^{-1}_{121}(\Sigma^*_{\{1,2\}\{1\}} \mathrm{f}_{y_{11}} \mathrm{f}_{y_{21}}) \subset \Sigma^*_{\{1,2\}\{1\}} \text{ and } \iota^{-1}_{ii'k} \in \mathscr{I}^{\{1,2\}\{1\}}_{\{i,i'\}\{k\}},$$

we now generally consider safety properties formalised by

$$\mathscr{F}^{\mathring{F}}_{IK} = \bigcup_{I' \subset I, K' \subset K, \iota^{\mathring{I}\mathring{K}}_{I'K'} \in \mathscr{I}^{\mathring{I}\mathring{K}}_{I'K'}} (\Pi^{IK}_{I'K'})^{-1}(\iota^{\mathring{I}\mathring{K}}_{I'K'}(\mathring{F})) \text{ and}$$

generated by $\mathring{F} \subset \Sigma^*_{\mathring{I}\mathring{K}}$.

For the privacy requirement above

$$\mathring{F} = \nu^{-1}_{121}(\Sigma^*_{\{1,2\}\{1\}} \mathrm{f}_{y_{11}} \mathrm{f}_{y_{21}}),\ \mathring{I} = \{1,2\},\ \mathring{K} = \{1\}.$$

By this definition

$$\mathscr{F}^{\mathring{F}}_{IK} = \emptyset \text{ for } |I| < |\mathring{I}| \text{ or } |K| < |\mathring{K}|, \tag{4}$$

as in that case $\mathscr{I}^{\mathring{I}\mathring{K}}_{I'K'} = \emptyset$ for each $I' \subset I$ and $K' \subset K$.

Now by the same argument as in our privacy example, we get

Theorem 2. *Self-similarity of $\mathscr{L}_{IK}$ implies that for a fixed $\mathring{F} \subset \Sigma^*_{\mathring{I}\mathring{K}}$ holds*

$$\mathscr{L}_{IK} \cap \mathscr{F}^{\mathring{F}}_{IK} = \emptyset \text{ for each index sets } I \text{ and } K$$

$$\text{iff for the fixed index sets } \mathring{I} \text{ and } \mathring{K} \quad \mathscr{L}_{\mathring{I}\mathring{K}} \cap \mathring{F} = \emptyset.$$

If $\mathscr{L}_{\mathring{I}\mathring{K}}$ and $\mathring{F}$ are regular subsets of $\Sigma^*_{\mathring{I}\mathring{K}}$ this can be checked by finite state methods [15]. On account of (4) it makes sense to consider safety properties defined by

$$\mathscr{F}_{IK} := \bigcup_{t \in T} \mathscr{F}^{\mathring{F}_t}_{IK} \text{ with finite } T \text{ and } \mathring{F}_t \subset \Sigma^*_{\mathring{I}_t\mathring{K}_t} \tag{5}$$

for each $t \in T$.

Definition 4 (Uniformly parameterised safety properties)**.** *Safety properties of the form* (5) *we call* uniformly parameterised.

Corollary 1. *For self-similar $\mathscr{L}_{IK}$ the parameterised problem of verifying a uniformly parameterised safety property is reduced to finite many fixed finite state problems if the corresponding $\mathscr{L}_{\mathring{I}_t\mathring{K}_t}$ and $\mathring{F}_t$ are regular languages.*

Example 3 can also be seen as a dependability example when we assume that the managed resource is a rail track, the clients are trains and the policy to ensure integrity and

safety of the system demands that only one train is allowed to use the rail track at a time. We can also derive an authentication example when we assume the client to be an ATM with f_x (authenticate credit card), f_y (draw-out cash) and f_z (eject credit card) and the server actions respectively manage a bank account. Please note that a stronger property "no further authentication before the card is ejected" does not hold for that specification.

VI. Conclusions and Future Work

The main result of this paper is to demonstrate the significance of self-similarity for verification of security properties. We assume that uniformly parameterised structures like the uniformly parameterised cooperations we have defined and used here are likely to appear in any highly scalable system or system of systems such as cloud computing platforms or vehicular communication systems.

It is well known that dynamic system properties such as dependability and security properties are divided into safety and liveness properties [14]. Safety properties can be formalised by formal languages as demonstrated in section V. We have shown here in particular that for self-similar $\mathscr{L}_{IK}$ the parameterised problem of verifying a uniformly parameterised safety property can be reduced to finite many fixed finite state problems under certain regularity restrictions. For abstractions defined by alphabetic language homomorphisms it is easy to see that an abstract system satisfies a safety property as considered in Sect. V iff the concrete system satisfies a corresponding safety property. So our notion of self-similarity is compatible with uniformly parameterised safety properties.

Concerning liveness properties ("Eventually something desired happens.") such a relation between abstract and concrete systems does not hold in general. In [16] a property of homomorphisms is given that implies a similar relation between liveness properties of an abstract and a concrete system w.r.t. a modified satisfaction relation ("Eventually something desired is possible."). Based on that framework we will investigate liveness aspects of uniformly parameterised cooperations as well as safety properties related to deadlocks in a forthcoming paper. Another topic of interest is the generalisation of this method to n-sided cooperations.

Acknowledgement

Roland Rieke developed the work presented here in the context of the project MASSIF (ID 257475) being co-funded by the European Commission within FP7.

References

[1] A. Fuchs and R. Rieke, "Identification of Authenticity Requirements in Systems of Systems by Functional Security Analysis," in *Workshop on Architecting Dependable Systems (WADS 2009), in Proceedings of the 2009 IEEE/IFIP Conference on Dependable Systems and Networks, Supplementary Volume*, 2009.

[2] C. N. Ip and D. L. Dill, "Verifying Systems with Replicated Components in Murφ," *Formal Methods in System Design*, vol. 14, no. 3, pp. 273–310, 1999.

[3] F. Derepas and P. Gastin, "Model checking systems of replicated processes with SPIN," in *Proceedings of the 8th International SPIN Workshop on Model Checking Software (SPIN'01)*, ser. Lecture Notes in Computer Science, M. B. Dwyer, Ed., vol. 2057. Toronto, Canada: Springer, May 2001, pp. 235–251.

[4] Y. Lakhnech, S. Bensalem, S. Berezin, and S. Owre, "Incremental Verification by Abstraction." in *TACAS*, ser. Lecture Notes in Computer Science, T. Margaria and W. Yi, Eds., vol. 2031. Springer, 2001, pp. 98–112.

[5] S. Basu and C. R. Ramakrishnan, "Compositional analysis for verification of parameterized systems," *Theor. Comput. Sci.*, vol. 354, no. 2, pp. 211–229, 2006.

[6] R. Milner, *Communication and Concurrency*, ser. International Series in Computer Science. NY: Prentice Hall, 1989.

[7] J. Bradfield and C. Stirling, "Modal logics and mu-calculi: an introduction," 2001. [Online]. Available: citeseer.ist.psu.edu/bradfield01modal.html

[8] P. Ochsenschläger and R. Rieke, "Abstraction Based Verification of a Parameterised Policy Controlled System," in *International Conference "Mathematical Methods, Models and Architectures for Computer Networks Security" (MMM-ACNS-7)*, ser. CCIS, vol. 1. Springer, September 2007.

[9] T. E. Uribe, "Combinations of Model Checking and Theorem Proving," in *FroCoS '00: Proceedings of the Third International Workshop on Frontiers of Combining Systems*. London, UK: Springer-Verlag, 2000, pp. 151–170.

[10] P. Ochsenschläger and R. Rieke, "Uniform Parameterisation of Phase Based Cooperations," Fraunhofer SIT, Tech. Rep. SIT-TR-2010/1, 2010. [Online]. Available: http://sit.sit.fraunhofer.de/smv/publications

[11] M. Jantzen, "Extending Regular Expressions with Iterated Shuffle," *Theor. Comput. Sci.*, vol. 38, pp. 223–247, 1985.

[12] J. Jedrzejowicz and A. Szepietowski, "Shuffle languages are in P," *Theor. Comput. Sci.*, vol. 250, no. 1-2, pp. 31–53, 2001.

[13] H. Björklund and M. Bojanczyk, "Shuffle Expressions and Words with Nested Data," in *Mathematical Foundations of Computer Science 2007*, 2007, pp. 750–761.

[14] B. Alpern and F. B. Schneider, "Defining liveness," *Information Processing Letters*, vol. 21, no. 4, pp. 181–185, October 1985.

[15] J. Sakarovitch, *Elements of Automata Theory*. Cambridge University Press, 2009.

[16] U. Nitsche and P. Ochsenschläger, "Approximately satisfied properties of systems and simple language homomorphisms," *Information Processing Letters*, vol. 60, pp. 201–206, 1996.